Lecture 4: Addressing modes

- **An instruction in the MC68000 contains two types of information**
  - The type of operation to be performed
  - The location of the operand(s) on which to perform the function: the **addressing modes**
- **The MC68000 supports 14 different addressing modes**
  - Data register direct
  - Address register direct
  - Absolute short
  - Absolute long
  - Register indirect
  - Post-increment register indirect
  - Pre-decrement register indirect
  - Register indirect with offset
  - Register indirect with index and offset
  - PC-relative with offset
  - PC-relative with index and offset
  - Immediate
  - Immediate quick
  - Implied register

Addressing modes summary

MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS <ea> GENERATION
Data register direct	D_n	<ea>=D_n
Address register direct	A_n	<ea>= A_n
Absolute short	xxx.S or <xxx	<ea>=(next word)
Absolute long	xxx.L or >xxx	<ea>=(next two words)
Register indirect	(A_n)	<ea>=(A_n)
Post-increment register indirect	(A_n)+	<ea>=(A_n), (A_n)←(A_n)+N
Pre-decrement register indirect	-(A_n)	(A_n)←(A_n)-N,<ea>=(A_n)
Register indirect with offset	d_{16}(A_n)	<ea>= (A_n)+d_{16}
Register indirect with index and offset	d_8(A_n,X_n)	<ea>= (A_n)+ (X_n)+d_8
PC-relative with offset	d16(PC)	<ea>= (PC)+d_{16}
PC-relative with index and offset	d_8(PC,X_n)	<ea>=PC+ (X_n)+d_8
Immediate	#data (32 bits)	Data=next 2 word
Immediate quick	#data (16 bits)	Data=next word
Implied register	CCR,SR,SP,PC	<ea>= CCR,SR,SP,PC

Data register direct

- **Description**
  - data register (D0-D7) is the source or destination of data
- **Example**
  - since '.B' is appended to MOVE, only the low byte of the destination data register is affected

INSTRUCTION	MOVE.B D0,D3			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE			D0 D3	10204FFF 1034F88A
AFTER			D0 D3	10204FFF 1034F8FF

Address register direct

- **Description**
  - address register (A0-A7) is the destination of data
  - only word or longword operands may be specified
  - a word operand is sign-extended to fit the register
- **Example**
  - The contents of A3 are copied onto A0

INSTRUCTION	MOVEA.L A3,A0			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE			A0 A3	00200000 0004F88A
AFTER			A0 A3	0004F88A 0004F88A

Absolute short

- **Description**
  - source or destination is a memory location whose address is specified in **one** extension word of the instruction
    - bits 16-23 of the full address are obtained by sign-extension of the 16-bit short address
- **Example**
  - source is immediate, destination is absolute short address
  - since operation is '.W', source is sign-extended to two bytes

INSTRUCTION	MOVE.W #$1E,$800			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE	000800 000801	12 34		
AFTER	000800 000801	00 1E		

Absolute long

- **Description**
  - source or destination is a memory location whose address is specified in **two** extension words of the instruction
- **Example**
  - source is immediate, destination is absolute long address
  - since operation is '.B', only one byte of memory is changed
- **Note**
  - Often, the distinction between abs-short and abs-long is transparent to the programmer due to the use of labels

INSTRUCTION	MOVE.B #$1E,$8F000			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE	08F000	FF		
AFTER	08F000	1E		

Register indirect

- **Description**
  - an **address register** contains the address of the source or destination operand
- **Example**
  - The instruction moves a longword stored in D0 to the memory location specified by the address in A0

INSTRUCTION	MOVE.L D0,(A0)			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE	001000 001001 001002 001003	55 02 3F 00	A0 D0	00001000 1043834F
AFTER	001000 001001 001002 001003	10 43 83 4F	A0 D0	00001000 1043834F

Post-increment register indirect

- **Description**
  - indicated by a '+' sign **after (A$_i$)**
  - **after** reading or writing data the address register is incremented by the number of bytes transferred
    - byte: [A$_i$]←[A$_i$]+1
    - word: [A$_i$]←[A$_i$]+2
    - longword: [A$_i$]←[A$_i$]+4

INSTRUCTION	MOVE.W (A5)+,D0			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE	001000 001001 001002 001003	45 67 89 AB	A5 D0	00001000 0000FFFF
AFTER	001000 001001 001002 001003	45 67 89 AB	A5 D0	00001002 00004567

Pre-decrement register indirect

- **Description**
  - indicated by a '-' sign **before** (A_i)
  - **before** reading or writing data the address register is decremented by the number of bytes transferred
    - byte: $[A_i] \leftarrow [A_i]+1$
    - word: $[A_i] \leftarrow [A_i]+2$
    - longword: $[A_i] \leftarrow [A_i]+4$

INSTRUCTION	MOVE.W D0,-(A7)			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE	001000 001001 001002 001003	10 12 83 47	A7 D0	00001002 00000143
AFTER	001000 001001 001002 001003	01 43 83 47	A7 D0	00001000 00000143

Register indirect with offset

- **Description**
  - a variation of register indirect that includes a 16-bit signed offset (displacement) as an extension word in the instruction
  - the sign-extended offset is added to the address register to form the effective address of the source or destination
- **Example**
  - effective address is 6 plus address register
  - value stored in the address register does not change

INSTRUCTION	MOVE.W 6(A0),D0			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE	001026 001027	07 BF	A0 D0	00001020 00000000
AFTER	001026 001027	07 BF	A0 D0	00001020 000007BF

Register indirect with index and offset

- **Description**
  - another variation of register indirect. An index register is used as well as an 8-bit signed offset
  - the effective address is formed by adding the sign-extended offset, the contents of the index register and the contents of the address register
- **Example**
  - <ea>=$10+$100A+$2=$101C

INSTRUCTION	MOVEA $10(A0,D0.L),A1			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE	00101C 00101D	EF 10	A0 A1 D0	0000100A 00000000 00000002
AFTER	00101C 00101D	EF 10	A0 A1 D0	0000100A FFFFEF10 00000002

PC-relative with offset

- **Description**
  - a 16-bit offset is added to PC to form effective address
  - only source operand can be addressed this way
  - this mode provides position-independent code
  - assembler computes offset by subtracting PC from label

```
                                   $0FFE=$2000-$1000-$2
00001000                              1    ORG      $1000
00001000  3A3A 0FFE                  2    MOVE.W   COUNT(PC),D5
00002000                              3    ORG      $2000
00002000  ABCD                        4    COUNT    DC.W     $ABCD
00002002                              5    END
```

INSTRUCTION	MOVE.W COUNT(PC),D5			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE	001026 001027	AB CD	PC D5	00001000 12345678
AFTER	001026 001027	AB CD	PC D5	00001004 1234ABCD

PC-relative with index and offset

- **Description**
  - an 8-bit signed offset plus an index register are used to compute the address relative to the PC

```
00001000                        1                ORG       $1000
00001000  303C 0005             2                MOVE.W    #5,D0
00001004  6100 0004             3                BSR       SQUARE
00001008  4E75                  4                RTS
0000100A  103B 0004             5       SQUARE   MOVE.B    TABLE(PC,D0.W),D0
0000100E  4E75                  6                RTS
00001010  000104091019          7       TABLE    DC.B      0,1,4,9,16,25
00001016                        8                END
```

INSTRUCTION	MOVE.B TABLE(PC,D0.W),D0			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE	001015	19	PC D0	0000100A ABCD0005
AFTER	001015	19	PC D0	0000100E 00000019

Immediate

- **Description**
  - immediate address uses **two** extension words to hold the source operand
  - data may be expressed in:
    - decimal (& prefix or none)
    - hexadecimal ($ prefix)
    - octal (@ prefix)
    - binary (% prefix)
    - ASCII (string within ‘’)

INSTRUCTION	MOVE.L #$1FFFF,D0			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE			D0	12345678
AFTER			D0	0001FFFF

Immediate quick

- **Description**
  - immediate quick addressing is an optimized case of immediate addressing whose binary code fits in one word (including the operand)
  - immediate operand is sign-extended to fit the 32-bit destination
  - available with the following instructions
    - MOVEQ (operand must be a 8-bit signed integer)
    - ADDQ (operand must be in the range 1 to 8)
    - SUBQ (operand must be in the range 1 to 8)

INSTRUCTION	MOVEQ #$1F,D0			
	MEMORY		REGISTER	
	ADDRESS	CONTENTS	NAME	CONTENTS
BEFORE			D0	12345678
AFTER			D0	0000001F